Data-Driven Rate Control for RDMA Networks: A Lightweight Online Learning Approach

Jiancheng Ye*, Dong Lin*, Kechao Cai†, Chao Zhou‡, Jianfei He§, and John C.S. Lui¶
*Network Technology Lab and Hong Kong Research Center, Huawei Technologies Co., Ltd.
†School of Electronics and Communication Engineering, Sun Yat-Sen University, China
‡Computing Product Line, Huawei Technologies Co., Ltd.
§Department of Computer Science, City University of Hong Kong
¶Department of Computer Science and Engineering, The Chinese University of Hong Kong
Email: yejiancheng@huawei.com, lindong10@huawei.com, caikch3@mail.sysu.edu.cn, ccc.zhou@huawei.com, jianfeihe2-c@my.cityu.edu.hk, cslui@cse.cuhk.edu.hk

2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS) | 979-8-3503-3986-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDCS57875.2023.00085

***Abstract*—Link speed in datacenter networks (DCNs) keeps growing rapidly, inducing an increasingly large portion of network flows to become short flows which can be finished within one round-trip time (RTT). This phenomenon makes many existing congestion control schemes ineffective because they iteratively adjust the sending rate based on the latest congestion feedback in multiple rounds. We find that the representative DCQCN scheme for RDMA exhibits substantial performance degradation when there are many short flows, and this is specially true in High Performance Computing (HPC) scenarios where most of Message Passing Interface (MPI) messages are small. In this paper, we propose a data-driven rate control framework which can learn from long-term online data about past rate control decisions via a lightweight online learning technique named Multi-Armed Bandit (MAB) which has a provable performance guarantee. Utilizing the framework, we devise a rate control scheme named Dolce-RC, which dynamically controls the rate increase and reduction by learning from online data. We implement Dolce-RC in commodity smart NICs, and show via testbed experiments and large-scale simulations that compared to DCQCN, Dolce-RC reduces average completion time of MPI messages by up to 68%, while not requiring any modification to switches.**

I. Introduction

In recent years, the link speed in datacenter networks (DCNs) has been growing rapidly. Meanwhile, more applications in DCNs require not only high throughput but also low latency. The Remote Direct Memory Access (RDMA) [1], [2] is a prevalent technology for modern DCNs. In RDMA, the networking protocol is implemented in network interface cards (NICs). RDMA can provide much higher throughput and lower latency via bypassing the host networking stack. To realize RDMA over Ethernet and IP networks, the RDMA over Converged Ethernet Version 2 (RoCEv2) [3] has been proposed. To facilitate a lossless Ethernet, RoCEv2 uses Priority-based Flow Control (PFC) [4] to prevent packet drops due to buffer overflow at the link layer.

It is important to emphasize that PFC causes several issues such as unfairness and victim flows [2], since it does not differentiate flows. To alleviate these issues, DCQCN [2] has been proposed as a flow-level congestion control protocol for RoCEv2. DCQCN is a rate-based congestion control scheme that takes advantage of both Quantized Congestion Notification (QCN) [5] and Data Center TCP (DCTCP) [6].

It has been reported that the representative DCQCN performs sufficiently well for long flows [2], [7]. But because the link speed in DCNs keeps growing, an increasing portion of flows will become short flows which can be finished within one or a few round-trip times (RTTs) [8], [9]. This case is particularly true for the High Performance Computing (HPC) applications which mostly use small Message Passing Interface (MPI) messages [10] for communication among nodes. For these small messages or short flows, they are usually sent with high data rates in a bursty manner. Consequently, even though the latest congestion signal (such as a Congestion Notification Packet (CNP) used by DCQCN) can be fed back to the sender, *it no longer serves as an effective indicator for adjusting the subsequent sending rates for these short flows because the sending process has been completed*. In this paper, we will show that a high ratio of short flows in the network traffic can significantly degrade the performance of DCQCN which needs to perform iterative tuning for the sending rate in multiple rounds. If the sending process only lasts for one or a few rounds, it is more important to determine appropriate initial sending rates for the short flows. *This issue will be more prominent with the trend of rapidly growing link speed in DCNs, since larger portion of flows will become short flows, inducing a fundamental challenge to existing congestion control schemes which rely on feedback-based iterative tuning.*

To address the aforementioned challenge, we propose an *online data-driven* rate control framework which does not rely on the latest congestion feedback, but rather makes use of long-term online data to make efficient rate control decisions. Here, long-term online data represents time-varying statistical information about the sender's actions and the corresponding feedbacks, which can be collected dynamically over time. We will show that the long-term online data can serve as better references for rate control because they provide more information about the effects of past decisions on the network. To make decisions by learning from the long-term online data, we design a lightweight online learning technique based on the Multi-Armed Bandit (MAB) theory [11]. Compared to deep

2575-8411/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDCS57875.2023.00085

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

learning and full reinforcement learning which exhibit high computational complexity, MAB is a lightweight and efficient learning framework with provable performance guarantee [11]. Thus, it is more suitable for making online real-time decisions in networking scenarios. There are some recent related works on applying online learning to congestion control. For example, PCC [12] and PCC Vivace [13] can learn from "micro-experiments" for TCP sending rates in an online fashion. In [14], MAB online learning is utilized to guide the selection of TCP initial windows. Different from these works, our work in this paper applies MAB to rate control in RDMA networks.

Contributions: In this paper, we propose a novel data-driven rate control framework for RDMA using a lightweight online learning approach. Our contributions are as follows:

- We show with real workloads that the representative DCQCN scheme for RDMA, which does not have learning capability and only relies on the latest congestion feedback to adjust the sending rate, becomes ineffective when there are many short flows, especially for HPC scenarios where applications mostly use small MPI messages.
- To address the aforementioned issue, we propose an online data-driven rate control framework which utilizes the long-term online data (instead of the latest feedback) to perform rate control for RDMA. To this end, we propose a novel MAB algorithm (i.e., DR-UCB) for adapting to and learning from the dynamic network environments. To the best of our knowledge, this is the first work that applies the lightweight MAB online learning to guide the rate control for RDMA based on long-term online data.
- We devise two rate control schemes (i.e., Dolce-RC and Dolce-RC+) that apply MAB to learn from online data, so as to make efficient rate control decisions for RDMA over dynamic networks. Dolce-RC is our first proposal that utilizes information carried in current protocols and extends an existing congestion control mechanism via smart NICs and thus can be deployed immediately, whereas Dolce-RC+ is a clean-slate design which explores extra information and can be deployed for future DCNs.
- We implement the proposed Dolce-RC using smart NICs, and provide details (including the challenges encountered and our solutions) for prototyping.
- We conduct both testbed experiments and large-scale ns-3 simulations, which comprehensively verify the superiority of the two proposed data-driven rate control schemes over the representative DCQCN scheme.

The rest of the paper is organized as follows. Section II gives a motivating example. In Section III, we propose our data-driven rate control framework. Section IV presents a specific rate control scheme named Dolce-RC. Section V presents both testbed and simulation results for evaluating Dolce-RC. Furthermore, we propose a clean-slate design called Dolce-RC+ in Section VI. Finally, we conclude the paper in Section VII.

II. A Motivating Example

This section provides a motivating example for showing that DCQCN which adjusts the sending rate based on the latest congestion signal will experience substantial performance degradation when there are many short flows in the network.

We consider a HPC scenario where nodes communicate with each other via small MPI messages. According to [10], most MPI messages in HPC networks are small, posing a real challenge to existing congestion control schemes. We set up a small testbed which consists of eight servers and three switches organized in a two-layer Clos topology. Four servers are connected to one leaf switch and the remaining four servers are connected to another leaf switch. The two leaf switches are connected to the same spine switch. More details about the testbed setup can be found in Section V-A. In order to accomplish a task, each of the eight server nodes periodically initiates short flows to send small *MPI_Alltoall* messages [10] to all the other server nodes. We choose the *MPI_Alltoall* collective operation because it is easier to generate congestion. Using these settings, we run the DCQCN as the congestion control scheme in the network. The resulting average latency (i.e., completion time) of the *MPI_Alltoall* collective operation for various message sizes within $[1\,\text{B}, 1024\,\text{KB}]$ is plotted in Fig. 5 (to save space, we directly use the results of the OSU Micro-Benchmarks experiment in Section V-B).

In Fig. 5, the values represented by the vertical and horizontal axes are displayed in a logarithmic scale. It can be seen that the average latency yielded by DCQCN (when PPN = 40) exhibits nonlinear increases for the message sizes within the range $[1024\,\text{B}, 128\,\text{KB}]$. The reason for this issue is that most of the transmissions for the small messages are completed within one or a few RTTs, making DCQCN's feedback-based iterative tuning mechanism ineffective. One can also observe that the average latency shows a sudden decrease and follows approximately linear increases for message sizes larger than 128 KB since DCQCN's iterative tuning starts to take effect. This example demonstrates that DCQCN does not perform well in the scenarios having many short flows, which are certain to happen when the link speed becomes higher.

III. The Data-Driven Rate Control Framework

In this section, we propose our general data-driven rate control framework and a novel MAB algorithm that can learn from and adapt to the dynamic network environments.

A. Proposed Data-Driven Rate Control Framework

An illustrative diagram of our data-driven rate control framework is given in Fig. 1. The essential idea for this framework can be explained as follows. The framework relies on long-term online data instead of the latest feedback for controlling the sending rate of a flow. The input of the framework can be various types of online data that serve as useful references for making rate control decisions. For example, the past sending rates, the corresponding Explicit Congestion Notification (ECN) marking (implied by CNPs), and the measured message completion times (MCTs) [8] are some of the useful and obtainable long-term online data. The framework maintains a small space (e.g., some space in smart NICs) for storing and updating the online data. To reduce the

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

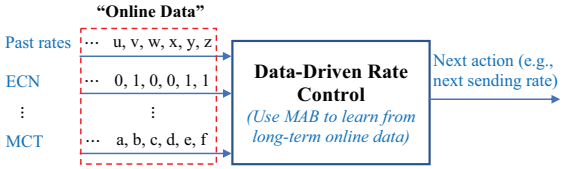


Fig. 1. A diagram of data-driven rate control.

size of this space, we can only keep track of some cumulative values of the online data, and this depends on the specific MAB algorithm used. Note that the online data used by the framework are not limited to the three types shown in Fig. 1. It is believed that the more useful online data it has, the better rate control decisions it can make. On the other hand, more online data incur more overheads for storing and processing. Thus, there exists a tradeoff between the amount of online data used and the optimality of the rate control decisions made, and our framework takes this tradeoff into consideration.

The core component of our framework is the MAB algorithm that is used to perform online learning from the online data. The framework does not stick to a unique MAB algorithm. In general, the MAB algorithm used by the framework should meet the following challenging requirements: 1) having a good learning performance measured by the regret [11], and 2) quickly adapting to the dynamic and non-stationary network environments. In the next subsection, we will present a novel MAB algorithm as an example which satisfies the above requirements. Finally, the output of the framework is simply the next action to take, such as the next sending rate for a new packet or message.

B. Proposed MAB Algorithm: DR-UCB

We now present our MAB algorithm that can learn from and adapt to the non-stationary network environments.

MAB is a lightweight online learning model for sequential decision making under uncertainty [11]. Suppose that there is a bandit machine with K arms (i.e., choices for different sending rates, rate control parameters, etc.). At each iteration, the decision maker chooses one arm and then receives a corresponding reward. The goal of the decision maker is to minimize the expected regret (i.e., the expected loss) over T iterations [15].

We realize that existing MAB algorithms cannot satisfy both the non-stationary and the fast adaptation requirements. Thus, we propose a novel MAB algorithm, called **D**iscounted and **R**andom Exploratory **UCB** (DR-UCB), which improves the discounted UCB algorithm [16] by incorporating a random exploration phase that enables the algorithm to make a balance between optimality and fast adaptation to changes.

Our DR-UCB algorithm is presented in Algorithm 1. Lines 3–5 of the algorithm correspond to the random exploration phase. In this phase, the algorithm selects each arm once for every $\lfloor \frac{K}{\epsilon} \rfloor$ iterations. This is equivalent to randomly explores the K arms with a small probability ϵ in the long run. This random exploration gives the algorithm the ability to quickly response to the environmental changes, but also

Algorithm 1 DR-UCB

Input: K arms (i.e., arm set $\mathcal{K} = \{0, 1, ..., K-1\}$), random exploration probability $\epsilon \in (0, 1)$, discount factor $\gamma \in (0, 1)$, constant $\xi > 0.5$.

1: **Initialize:** $n_i(0) = 0$ for $i = 0, 1, ..., K-1$.
2: **for** $t = 0, 1, ..., T$ **do**
3: $r \leftarrow (t \mod \lfloor \frac{K}{\epsilon} \rfloor)$.
4: **if** $r < K$ **then**
5: $I_t = r$.
6: **else**
7: $I_t = \arg\max_{i \in \mathcal{K}} \frac{\sum_{\tau=0}^{t-1} \gamma^{t-1-\tau} X_i(\tau) \mathbb{1}(I_\tau = i)}{n_i(t-1)} + \sqrt{\frac{\xi \log \sum_{i=0}^{K-1} n_i(t-1)}{n_i(t-1)}}$.
8: **end if**
9: Select arm I_t and observe reward $X_{I_t}(t)$.
10: $n_i(t) \leftarrow \sum_{\tau=0}^{t} \gamma^{t-\tau} \mathbb{1}(I_\tau = i)$, for all $i \in \mathcal{K}$.
11: **end for**

induces a small loss (up to ϵ) in optimality. Lines 6–10 perform the learning process similar to the discounted UCB algorithm. Specifically, Line 7 indicates that at each iteration t, the algorithm chooses an arm I_t that has the highest upper confidence bound for the expected reward. On the right-hand side of the equation in Line 7, the first term corresponds to the discounted average reward of arm i, where γ is the discount factor, $X_i(\tau)$ denotes the instantaneous reward of arm i at iteration τ, $n_i(t-1)$ is the discounted number of times arm i has been chosen by iteration $(t-1)$, and $\mathbb{1}(I_\tau = i)$ equals to one if $I_\tau = i$ (it is zero otherwise). The second term can be considered as the discounted exploration bonus, which is related to the confidence interval for the expected reward. Once the arm I_t is determined for iteration t, the algorithm will select it and observe its instantaneous reward $X_{I_t}(t)$, as shown in Line 9. Finally, the discounted number of times an arm i has been chosen so far will be updated in Line 10. The DR-UCB algorithm is simple enough so that it can be utilized and implemented by smart NICs (see Section IV-B).

Next, we present a theorem that guarantees a good regret upper bound for the DR-UCB algorithm.

Theorem 1. *The DR-UCB algorithm guarantees the regret upper bound:* $(1-\epsilon)O(\sqrt{T})+\epsilon T$, *where* T *denotes the number of iterations and* ϵ *is a small random exploration probability.*

Proof. A detailed proof is given in Appendix A. □

Theorem 1 implies that the average regret (i.e., $\frac{1}{T}$regret) at most approaches to a small ϵ as the number of iterations T goes to infinity. This means that DR-UCB performs nearly as well as the oracle (up to a small gap ϵ) on the average. This small gap is due to the tradeoff made between optimality and fast adaptation to environmental changes. Thus, ϵ should be set small enough (e.g., 1%) such that a good balance between performance loss and timely response can be achieved.

IV. The Dolce-RC Scheme

This section presents a specific rate control scheme based on our data-driven rate control framework. The proposed scheme

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

is named ***D**ata-driven **o**nline **l**earning **c**ongestion-awar**e** **R**ate **C**ontrol (Dolce-RC)*, which dynamically and intelligently tunes two important parameters α and β controlling the rate changes, via learning from long-term online data.

A. Design of Dolce-RC

We now describe our design rationale and the Dolce-RC scheme in detail. In general, almost all congestion control schemes have to design how to increase and decrease the sending rate of a sender. The well-known additive increase/multiplicative decrease (AIMD) mechanism is a prevalent concept used by many congestion control schemes for adjusting the sending rate. We follow this general concept and introduce two important parameters α and β to realize a general AIMD behavior in the Dolce-RC scheme.

Recall that as a widely deployed congestion control scheme for RDMA, DCQCN performs quite well for long flows, but experiences significant performance degradation when there are many short flows in the network. To facilitate easy deployment for Dolce-RC, we take advantage of the DCQCN architecture but make substantial modifications to it so as to overcome its issues. Note that DCQCN maintains a rate reduction factor, α, which controls the rate reduction when receiving a CNP. It derives from DCTCP [6] and needs to be tuned iteratively based on two equations (i.e., equations (1) and (2) in [2]) so as to converge to an appropriate value [7]. It cannot function well in networks containing many short flows. In Dolce-RC, we use the same symbol α to represent the parameter regulating the rate reduction, but we do not utilize the two equations for updating α since they are inappropriate for short flows. Instead, we directly provide several predetermined values for α as components of the discrete arms in our MAB framework.

On the other hand, we introduce another parameter β in Dolce-RC, which regulates the rate increase during the fast recovery phase by controlling the relation between the current rate R_C and the target rate R_T. Note that R_T serves as an upper bound for R_C in DCQCN. We maintain this functionality and further use it to control the rate increase for different flows in Dolce-RC. Specifically, the parameter β is incorporated into the equation of updating R_T upon receiving a CNP in Dolce-RC, as shown below:

$$R_T = \beta R_C, \tag{1}$$

where β can either be within the range $[0.5, 1]$, or slightly larger than one. In particular, (1) becomes the same as that used by DCQCN when $\beta = 1$.

After the update of R_T when receiving a CNP, R_C is reduced according to the following equation in Dolce-RC:

$$R_C = (1 - \alpha/2)R_C, \tag{2}$$

where α is within the range $0 \le \alpha \le 1$. Unlike DCQCN, here the value of α is directly selected from a set of predetermined values without using iterative tuning.

The rate evolution of Dolce-RC controlled by the two parameters α and β is illustrated in Fig. 2. When receiving a CNP, the target rate R_T is updated using (1). Then, the current rate R_C is updated based on (2), where the value of α is selected from a set of given values. We use the red solid line and the black solid line to represent the evolutions of R_T (when $\beta = 1$) and the corresponding R_C, respectively. In contrast, the blue dash line shows the evolution of R_C when $\beta < 1$. It can be seen that upon finishing the fast recovery phase, R_C is smaller than that of the black solid line if $\beta < 1$, which means that the recovery of R_C is slowed down. On the contrary, the red dash line corresponds to the evolution of R_C when $\beta > 1$. In this case, R_C grows faster than that of the black solid line.

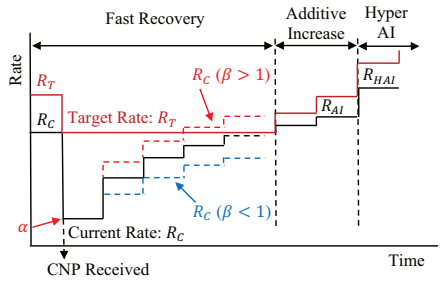


Fig. 2. Rate evolution of Dolce-RC, where α (with given values) is a parameter controlling the reduction in the current rate R_C when receiving a CNP, and β is a parameter regulating the rate increase by controlling the relation between R_C and the target rate R_T.

In Dolce-RC, the values of α and β are dynamically selected from a given arm set (i.e., action space) for every connection[1]. To this end, our DR-UCB algorithm is used to "learn" proper values of α and β from long-term online data.

In order to apply the MAB online learning framework to the rate control scenario, the key is to properly define the arms and the corresponding reward function. Next, we give the definitions about arms and reward function in Dolce-RC.

Definition 1 (*Arms in Dolce-RC*)**.** *In Dolce-RC, an arm is a specific action for setting α and β. An arm set is defined as $\mathcal{K} = \{\{\alpha_0, \beta_0\}, \{\alpha_1, \beta_1\}, ..., \{\alpha_{K-1}, \beta_{K-1}\}\}$ (with K arms), where arm i ($i = 0, 1, ..., K-1$) consists of two constants α_i and β_i, which correspond to the values of α and β, respectively.*

To explain how we determine the arm set in Dolce-RC, let us consider the DCQCN model incorporated with (1) for updating R_T. We revise the original DCQCN fluid model [7] to derive the relation between α and β in Dolce-RC, so as to determine a proper arm set.

Here, we provide a theorem that gives the relation between α and β when the Dolce-RC system is in the steady state.

Theorem 2. *For Dolce-RC that modifies DCQCN by incorporating with* (1) *and ignoring the two related equations of updating the rate reduction factor, the two parameters $\alpha^{(i)}$ and $\beta^{(i)}$ associated with a connection i satisfy the following*

[1]For the flows associated with a specific connection, a single set of α and β is maintained for them.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

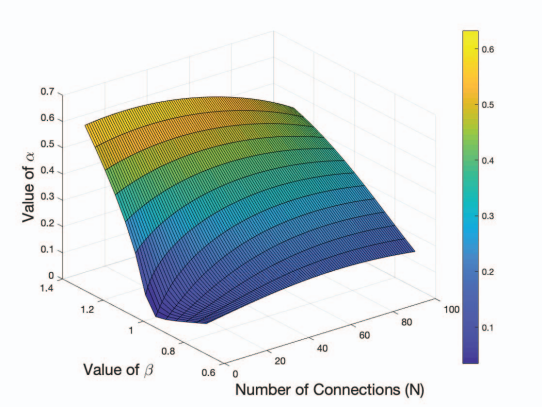


Fig. 3. Relations among α, β, and N in Dolce-RC.

relation in the steady state:

$$\alpha^{(i)} = \frac{\beta^{(i)} - 1 + \frac{\tau R_{AI}}{a}(c+e)}{\frac{a}{\tau R_C^{(i)*}(b+d)}}, \tag{3}$$

where τ is the CNP generation timer, R_{AI} is a fixed rate increase step, $R_C^{(i)*}$ represents the steady-state current rate of connection i, a, b, c, d, and e are auxiliary variables whose expressions are the same as equation (12) in [7].

Proof. A detailed proof is given in Appendix B. □

In particular, (3) is reduced to the equation (11) for DCQCN in [7] when $\beta^{(i)} = 1$. According to [7], $R_C^{(i)*}$ can be estimated by $R_C^{(i)*} = \frac{C}{N}$, where C is the bottleneck link capacity and N is the number of connections traversing the bottleneck link. Thus, from (3), one can compute the steady-state value of α, given the values of β and N. For example, we plot a graph showing the relations among α, β, and N based on (3) in Fig. 3, which uses the bottleneck link capacity $C = 100$ Gbps, $\beta \in [0.7, 1.3]$, and the number of connections $N \in [1, 100]$. It can be observed that: 1) for a given number of connections N, α increases with β, and 2) for a given value of β, α does not change too much and appears to have a maximum value, when N increases. Considering a practical range of N, we can choose proper values for α and β based on this relation graph, so as to construct the arm set for Dolce-RC.

Definition 2 (*Reward function in Dolce-RC*)**.** *In Dolce-RC, the reward function for arm i at iteration t is given by:*

$$X_i(t) = 1 - \frac{num\_ECNmarked\_packets(i,t)}{num\_packets\_sent(i,t)}, \tag{4}$$

where $X_i(t) \in [0,1]$ represents the instantaneous reward for arm i at iteration t, num_packets_sent(i, t) denotes the total number of packets sent using arm i during the interval between two received CNPs, and num_ECNmarked_packets(i, t) is the number of ECN-marked packets using arm i during the same interval. Here, the interval between two CNPs received by the sender needs to be at least five times of the CNP interval (i.e., about one fast recovery phase) in order to have a more accurate estimation about the effect of the chosen arm.

Algorithm 2 Dolce-RC

Input: Arm set $\mathcal{K} = \{\{\alpha_0, \beta_0\}, \{\alpha_1, \beta_1\}, ..., \{\alpha_{K-1}, \beta_{K-1}\}\}$, $\epsilon \in (0,1)$, $\gamma \in (0,1)$, and $\xi > 0.5$.

1: **for** every qualified CNP indexed by $t = 0, 1, ...,$ **do**
2: Measure the reward $X_{I_t}(t)$ using (4).
3: Update $n_i(t)$ for all arm $i \in \mathcal{K}$, according to Line 10 of DR-UCB.
4: $t \leftarrow t + 1$.
5: Execute Lines 3–8 of DR-UCB to obtain a suggested decision I_t (i.e., arm I_t) for iteration t.
6: Set $\alpha = \alpha_{I_t}$ and $\beta = \beta_{I_t}$.
7: Run the revised DCQCN scheme with α and β.
8: **end for**

Note that in reality, the reward evaluating a chosen arm is usually delayed for some time due to network latency. To tackle this practical issue, we measure a more long-term ECN ratio of $\frac{num\_ECNmarked\_packets(i,t)}{num\_packets\_sent(i,t)}$ during a reasonable period, as shown in (4). The design goal for the reward function of Dolce-RC is to optimize the aforementioned ECN ratio by balancing the total number of packets sent and the number of ECN-marked packets. However, it is challenging to find a good balance in the dynamic network environments. We will show that combining with MAB online learning, the defined reward function can significantly help reduce MCT and improve throughput since it leads to using the switch buffer more efficiently, especially when there are many short flows.

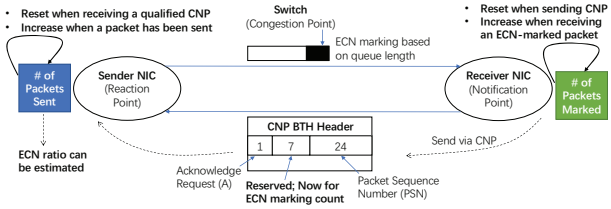
We summarize the Dolce-RC scheme in Algorithm 2. Dolce-RC introduces α and β to control the rate adjustments on a per-CNP basis. Compared to DCQCN, Dolce-RC has made substantial modifications in two aspects: 1) introducing β to update R_T via (1), and 2) removing the two equations for iteratively updating α but using MAB online learning to directly select a proper value for α. The values of α and β are dynamically selected from a predefined arm set.

In Algorithm 2, for every qualified CNP indexed by t, the reward $X_{I_t}(t)$ associated with the recently chosen arm I_t is updated using (4). A qualified CNP means that the interval between the last qualified CNP and the newly received one should be at least one fast recovery phase. Dolce-RC only uses the easily obtainable ECN information as the long-term online data. Then, $n_i(t)$ for all arm $i \in \mathcal{K}$ are updated according to DR-UCB. Lines 2–3 actually measure the effect of the recently chosen arm. Next, t is increased by 1, and then Lines 3–8 of DR-UCB are invoked to find a proper arm I_t for this iteration. Finally, α and β are updated to α_{I_t} and β_{I_t} respectively, and the revised DCQCN is run with the updated α and β.

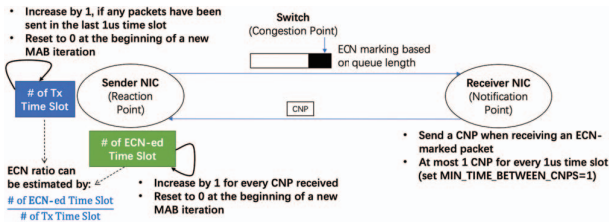
B. Implementation for Dolce-RC via Smart NICs

We implement a prototype of the proposed Dolce-RC in commodity smart NICs. Dolce-RC is a NIC based scheme and does not require any modifications to the switches. When implementing Dolce-RC in smart NICs, there are several challenges that need to be addressed. In this subsection, we provide related details for the implementation of Dolce-RC.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.



(a) Ideal case for estimating ECN ratio.



(b) Actual case for estimating ECN ratio via commodity smart NICs.

Fig. 4. Procedure for estimating ECN ratio using smart NICs.

1) Estimation of ECN Ratio via CNPs. Recall that Dolce-RC uses ECN related information to evaluate a chosen arm (i.e., action) via the reward function (4). Computing the reward of a chosen arm requires the estimation of the ECN ratio $\frac{num\_ECNmarked\_packets(i,t)}{num\_packets\_sent(i,t)}$. In the DCQCN architecture, a CNP packet does not provide such information. Therefore, we come up with an effective method to estimate the ECN ratio.

Fig. 4 illustrates a specific approach for estimating the ECN ratio using smart NICs. We first present an ideal case where the ECN ratio can be estimated accurately, as shown in Fig. 4(a). However, since the ideal case is currently not supported by mainstream commodity smart NICs, we further present an actual case for estimating the ECN ratio in Fig. 4(b), which is actually utilized by this paper.

Ideal case: At the sender's NIC, the number of packets sent using each arm i is maintained and increased by one when a packet has been sent using this arm. At the receiver's NIC, the number of ECN-marked packets for each arm i is maintained and increased by one when an ECN-marked packet has been received for this arm. Moreover, the 7-bit reserved field in the CNP BTH header [3] is used to store the number of ECN-marked packets, so that this information can be fed back to the sender via a CNP. After receiving a qualified CNP, the sender's NIC can estimate the ECN ratio via dividing the number of ECN-marked packets by the number of packets sent.

Actual case: Since mainstream commodity smart NICs (such as Mellanox ConnectX series) currently do not support the modification of the CNP header, we use another approach to estimate the ECN ratio. Consider every 1 μs as a distinct time slot and define a Tx time slot as the time slot in which there is a data transmission. The sender's NIC can compute the number of Tx time slots. At the receiver's NIC, when an ECN-marked packet is received and the minimum CNP interval (i.e., 1 μs, which can be set via the NIC's parameter MIN_TIME_BETWEEN_CNPS = 1) is satisfied, a new CNP packet will be sent to the sender. The sender's NIC also maintains the number of received CNPs and treats it as the number of ECN-marked time slots. The ECN ratio can be estimated via dividing the number of ECN-marked time slots by the number of Tx time slots.

2) Limited Storage at NICs. Commodity smart NICs usually have storage limitation. To implement Dolce-RC in smart NICs, we take the storage limitation into consideration and come up with a feasible solution. Specifically, Dolce-RC only requires four bytes of storage per arm, and the arm set for a connection has four arms only. Thus, each connection only uses 20 bytes (additional four bytes are used for recording the ECN ratio and iterations) of storage.

We now explain why four bytes are sufficient for each arm. Referring to DR-UCB (Algorithm 1), there are two important variables for each arm i: 1) the discounted accumulative reward $\sum_{\tau=0}^{t-1} \gamma^{t-1-\tau} X_i(\tau) \mathbb{1}(I_\tau = i)$ in Line 7, which is denoted by *RewardACC* here, and 2) the discounted number of chosen times $\sum_{\tau=0}^{t} \gamma^{t-\tau} \mathbb{1}(I_\tau = i)$ in Line 10, which is denoted by *ArmTriedNum* here. We use four bytes (a DWORD) to represent each of these two variables in any computations involving them. However, we only use two bytes to store each of them in NICs. To achieve this, we need to perform a few additional steps (such as shifting and masking) when loading and storing them. Due to space limitation, we do not describe the details here, but give the reasons why it is feasible to do so as follows. First, the discount factor $\gamma < 1$ ensures that both *RewardACC* and *ArmTriedNum* have bounded maximum values, which enable representations for them using fixed number of bits. Second, by estimating the possible ranges for *RewardACC* and *ArmTriedNum*, we can discard some lower and higher bits from their DWORD representations without significantly affecting the accuracy.

3) Fixed-Point based Approximations. In the DR-UCB algorithm used by Dolce-RC, there are several floating-point operations. In particular, logarithms need to be performed in Line 7. Since many commodity smart NICs only support fixed-point operations, we utilize fixed-point based approximations for floating-point operations. Due to space limitation, we do not describe the details here. Instead, we give a brief summary as follows. One can make use of the ideas proposed in [17], [18] to approximate floating-point logarithms with fixed-point operations. A public available implementation of a fixed-point logarithm algorithm in *C* can be found in [19]. Similarly, one can refer to related works in the literature such as [20] for the fixed-point implementations for square root and reciprocal.

V. Performance Evaluation

In this section, we conduct both testbed experiments and ns-3 [21] simulations, so as to evaluate the performance of the proposed Dolce-RC scheme. We compare Dolce-RC with the representative DCQCN scheme[2] in different scenarios in terms of important performance metrics. HPCC [23] is another popular rate control scheme for RDMA. However, in terms of fair comparison, we do not include it since it utilizes more precise in-network telemetry (INT) information.

[2]We utilize the public available source codes of DCQCN from [22].

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

A. Evaluation Setup

First, we provide specific evaluation setup for both testbed experiments and large-scale simulations as follows.

Testbed: We have built a small testbed which consists of eight servers and three switches. They are organized in a two-layer Clos topology. Specifically, four servers are connected to one leaf switch and the remaining four servers are connected to another leaf switch. Each of the two leaf switches is connected to the same spine switch via four independent 100 Gbps links. Each server has a smart NIC with 100 Gbps, an Intel Xeon Gold 6248 Processor with 20 cores and 40 threads, and an NVIDIA GPU. Our proposed Dolce-RC is implemented in the smart NICs. Each link has 1 μs propagation delay, and thus the base RTT is 4 μs within a rack and 8 μs across racks.

Simulator (ns-3): The topology for ns-3 simulations is a three-layer fat-tree with a much larger scale compared to the testbed experiments. Specifically, there are 1024 servers, 128 ToR switches, 128 aggregation switches, and 64 core switches. Each ToR switch is connected by 8 servers, and each spine switch is connected by 16 aggregation switches. Each server has a 200 Gbps NIC. All links have the capacity of 200 Gbps and 1 μs propagation delay. Thus, the minimum base RTT is 4 μs and the maximum base RTT is 12 μs.

Workloads: For HPC scenarios, we utilize the public available OSU Micro-Benchmarks [24] which measures the performance of an MPI implementation and VASP [25] as a realistic HPC application for further evaluation. For DCN scenarios, we use the widely acceptable realistic workloads Cache Follower [26] and Data Mining [27], where the majority of the flow sizes are less than 1 MB. In addition, we employ two types of average network loads, namely 30% and 50%, which can be achieved by properly adjusting the flow generation rate.

Parameter settings: For DCQCN and Dolce-RC, we set the ECN-marking thresholds to K_{min} = 7 KB and K_{max} = 500 KB for fair comparisons. We use the dynamic PFC with the headroom of 330 packets (assuming the packet size is 1000 B). The message sizes used in HPC scenarios are within the range [1 B, 1024 KB]. For the DR-UCB algorithm used by Dolce-RC, we use the discount factor $\gamma = 0.9998$, random exploration probability $\epsilon = 0.015$, and $\xi = 0.5001$.

B. Testbed Experiments

We present two scenarios for comparing the proposed Dolce-RC with DCQCN in the testbed experiments.

1) OSU Micro-Benchmarks Experiment. As aforementioned, the OSU Micro-Benchmarks measures the performance of an MPI implementation. Using this benchmark (version 5.8), we evaluate the performance of Dolce-RC in the HPC environment. Specifically, the eight servers in our testbed configure 16 or 40 processes per node (PPN). A larger PPN implies that the network becomes more congested. We use the *MPI_Alltoall* Latency Test [24] to measure the average latency of the *MPI_Alltoall* collective operation across multiple processes for various message sizes within [1 B, 1024 KB].

Fig. 5 presents the average latency achieved by Dolce-RC and DCQCN in this benchmark test. Note that the values

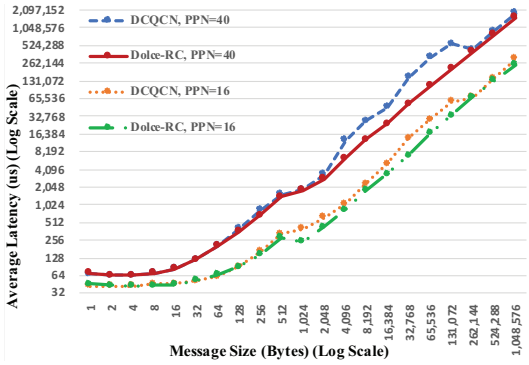


Fig. 5. Testbed results via OSU Micro-Benchmarks.

represented by the vertical and horizontal axes are displayed in a logarithmic scale. It can be seen that for the message sizes within the range [1024 B, 128 KB], the average latency of the *MPI_Alltoall* operation achieved by Dolce-RC is much lower than that achieved by DCQCN. The performance gain becomes larger when PPN is increased from 16 to 40. In particular, comparing to DCQCN, Dolce-RC yields the maximum reduction of 68% (for message size of 64 KB and PPN = 40) in the average latency achieved.

2) VASP Application Experiment. We further run the widely used HPC application, VASP [25] (version 5.4.4), in our testbed equipped with NVIDIA GPUs, so as to evaluate Dolce-RC in realistic HPC environment. We compare Dolce-RC with DCQCN in terms of overall job completion time (JCT) including both communication and computing aspects, using an NVIDIA's GPU VASP benchmark [28] with PPN = 12. In this experiment, DCQCN yields a JCT of 139.086 s, whereas Dolce-RC achieves a lower JCT of 133.863 s (a reduction of 3.76% after taking both communication and computing into account). This further corroborates the advantage of Dolce-RC over DCQCN in real HPC scenarios.

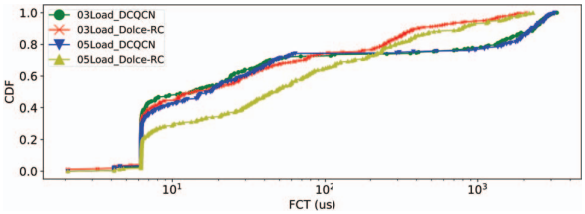
C. Large-Scale Packet-Level Simulations

This subsection presents large-scale ns-3 simulations to further evaluate the performance of Dolce-RC in DCN scenarios. There are two test scenarios using 1) the Cache Follower workload [26], and 2) the Data Mining workload [27].

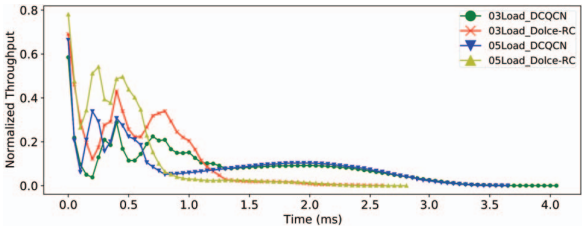
1) Cache Follower. We compare Dolce-RC with DCQCN in a test scenario using the Cache Follower realistic workload, where 71% of the flow sizes are less than 1MB and the average flow size is 701 KB [26]. Using the settings described for simulator in Section V-A with the Cache Follower workload and two types of network loads (30% and 50%), we run Dolce-RC and DCQCN in the network, respectively. Fig. 6 presents the flow completion time (FCT) distributions of all flows and the evolutions of aggregate throughput achieved by the two schemes. Table I further gives the mean FCT comparison.

It can be seen in Fig. 6(a) that Dolce-RC completes all flows earlier than DCQCN across different network loads, and it also yields much lower mean FCT as shown in Table I. In addition, the 99th-percentile FCT of Dolce-RC for 30%

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.



(a) FCT distributions.



(b) Evolutions of aggregate throughput.

Fig. 6. Test scenario for Cache Follower.

TABLE I
MEAN FCT COMPARISON USING CACHE FOLLOWER

Scheme	30% Network Load	50% Network Load
DCQCN	481.43 μs	546.48 μs
Dolce-RC	168.01 μs	234.97 μs

TABLE II
MEAN FCT COMPARISON USING DATA MINING

Scheme	30% Network Load	50% Network Load
DCQCN	1509.33 μs	1465.95 μs
Dolce-RC	1288.00 μs	1250.19 μs

and 50% loads are 1829.14 μs and 2024.28 μs, respectively, whereas the 99th-percentile FCT of DCQCN for 30% and 50% loads are 3088.51 μs and 3074.44 μs, respectively. Thus, Dolce-RC significantly outperforms DCQCN in terms of the mean and 99th-percentile FCT across different loads. Fig. 6(b) shows that Dolce-RC achieves higher throughput during the early stage of the simulation, and then the throughput is reduced later because most of the flows have been finished.

2) Data Mining. We conduct the second test using the Data Mining realistic workload, where 91% of the flow sizes are less than 1 MB and the average flow size is 7.41 MB [27]. The mean FCT comparison for this case is presented in Table II, where Dolce-RC also yields lower mean FCT than DCQCN.

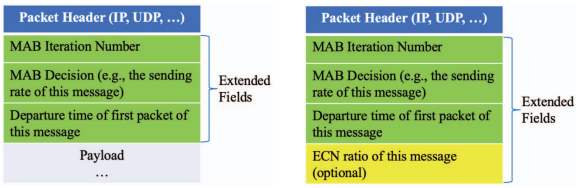
VI. A CLEAN-SLATE DESIGN: DOLCE-RC+

In this section, we further propose a clean-slate design for data-driven rate control, which demonstrates a purely data-driven learning-based new congestion control paradigm and is expected to be deployed for next generation of DCNs.

A. The Dolce-RC+ Scheme

We name the clean-slate design Dolce-RC+, which strictly follows the data-driven rate control framework in Fig. 1.

Different from Dolce-RC, Dolce-RC+ moves away from the DCQCN architecture for rate evolution. Recall that the data-driven rate control framework in Fig. 1 takes in several types



(a) Data packet. (b) ACK packet.

Fig. 7. Data and ACK packets in Dolce-RC+.

of online data and then directly determines the next sending rate using MAB. In Dolce-RC+, the past sending rates, the corresponding measured MCTs, and the ECN information are all utilized as long-term online data for learning. We expect that future commodity smart NICs can support the measurement of MCTs, so that Dolce-RC+ can be implemented to realize purely data-driven congestion control for future DCNs.

Definition 3 (*Arms in Dolce-RC+*)**.** *In Dolce-RC+, an arm is a specific action for setting the next sending rate. An arm set is defined as* $\mathcal{K} = \{r_0, r_1, ..., r_{K-1}\}$*, where* $r_i = \frac{1}{2^{K-1-i}} \cdot C$ *is the sending rate associated with arm* i *and* C *is the link capacity.*

Here, we define the discrete arms of Dolce-RC+ as the link capacity divided by different powers of two. In addition, the rates (i.e., r_i) represented by arms are in ascending order.

Definition 4 (*Reward function in Dolce-RC+*)**.** *In Dolce-RC+, the reward function for arm* i *at iteration* t *is given by:*

$$X_i(t) = \frac{ideal\_MCT(t)}{actual\_MCT(i,t)} \cdot \Big(1 - ECN\_Ratio\,(i,t)\Big)^i, \quad (5)$$

where $X_i(t) \in [0,1]$ *represents the instantaneous reward for arm* i*,* $actual\_MCT(i,t)$ *is the actual MCT of the message* t *sent using arm* i *(i.e.,* r_i*),* $ideal\_MCT(t)$ *is the ideal MCT for message* t*, and* $ECN\_Ratio\,(i,t)$ *is the ECN ratio.*

Compared to (4), we incorporate a new MCT-related factor into the reward function of Dolce-RC+. In (5), $\frac{ideal\_MCT(i,t)}{actual\_MCT(t)}$ equals to one over the slowdown [29] of the message. Choosing a higher sending rate may somewhat help reduce the actual MCT of the message, but may also result in a smaller value of $(1 - ECN\_Ratio\,(i,t))$. We use the factor $\big(1 - ECN\_Ratio\,(i,t)\big)^i$ to balance the reduction of MCT and the increase of queue length, such that both the MCT of the message and the buffer occupancy of the switch can be optimized. Since rates are arranged in ascending order in the arm set, we use the arm index i as an exponent to gradually increase the difficulty in selecting higher rates.

Intuitively, the design goal for the reward function of Dolce-RC+ is to reduce the MCTs of messages while keeping the queue length relatively low by selecting appropriate sending rates. MCT is crucial for application performance, but the optimization of it is not straightforward because it is jointly affected by several factors such as the sending rate, the queue length and the scheduling policy at the switch.

The proposed Dolce-RC+ scheme is summarized in Algorithm 3. Unlike Dolce-RC, the rate selection in Dolce-RC+ is

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3 Dolce-RC+

Input: Arm set $\mathcal{K} = \{r_0, r_1, ..., r_{K-1}\}$, where the sending rate $r_i = \frac{1}{2^{K-1-i}} \cdot C$, $\epsilon \in (0,1)$, $\gamma \in (0,1)$, and $\xi > 0.5$.

1: **for** every received ACK indexed by $t = 0, 1, ...,$ **do**
2: Measure the reward $X_{I_t}(t)$ using (5).
3: Update $n_i(t)$ for all arm $i \in \mathcal{K}$, according to Line 10 of DR-UCB.
4: $t \leftarrow t + 1$.
5: Execute Lines 3–8 of DR-UCB to obtain a suggested decision I_t (i.e., arm I_t) for iteration t.
6: Use rate r_{I_t} to send a new message indexed by t.
7: **end for**

triggered by every received acknowledgement (ACK) packet and on a per-message basis. Specifically, upon receiving an ACK packet acknowledging the receipt of a whole message t, the reward $X_{I_t}(t)$ of arm I_t (i.e., rate r_{I_t}) used for sending this message is updated using (5). To facilitate the measurement of MCT, we make some changes to the data packet and ACK packet for Dolce-RC+, which are described in Fig. 7. We insert three extended fields into the data packet, as shown in Fig. 7(a). Specifically, the first field "MAB Iteration Number" corresponds to iteration t in Line 5 of Algorithm 3. The second field "MAB Decision" indicates the specific decision made by MAB for this iteration, namely, the selected sending rate r_{I_t} for a new message t to be sent. The third field specifies the departure time of the first packet of this message. Upon receiving the last packet of a message, the receiver will send an ACK packet to the corresponding sender. We insert four extended fields into the ACK packet, as can be seen in Fig. 7(b). The first three extended fields are the same as that of the data packet. In addition, the receiver may optionally compute the ECN ratio of the message and place it into the fourth extended field. Once the sender receives the ACK packet, it can measure the actual MCT of the message by computing the difference between the receiving time of the ACK and the departure time of the first packet of this message.

We summarize the workflow in Dolce-RC+ as follows:

1) The sender sends data packets of a message t to the receiver using rate r_{I_t} determined by Dolce-RC+.

2) The switch may set the ECN bits of the packets traversing it, depending on its ECN-marking policy.

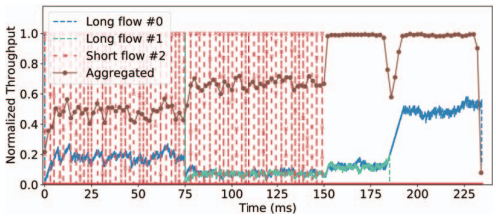
3) Upon receiving the last packet of message t, the receiver will construct an ACK and send it to the sender.

4) When the sender receives the ACK acknowledging message t, it will perform Lines 2–6 of Algorithm 3.

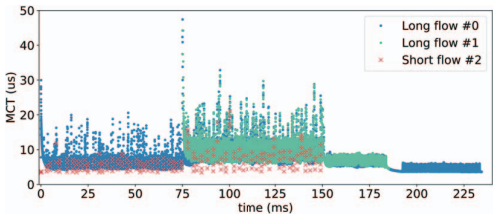
B. Evaluation of Dolce-RC+

In this subsection, we present preliminary ns-3 simulation results to demonstrate the performance of Dolce-RC+. We compare Dolce-RC+ with DCQCN in a dynamic network scenario, where flows join and leave dynamically over time.

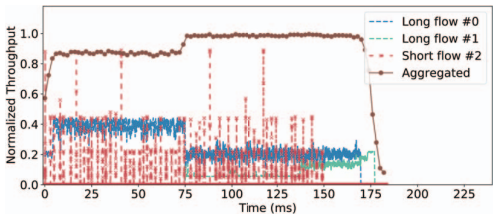
To easily evaluate Dolce-RC+ in a dynamic environment, we use a dumbbell topology for this scenario where a sender i ($i = 1, 2, ..., 32$) transmits data to the corresponding receiver



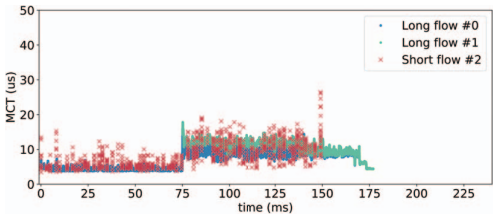
(a) Evolutions of throughput by DCQCN.



(b) Evolutions of MCT by DCQCN.



(c) Evolutions of throughput by Dolce-RC+.



(d) Evolutions of MCT by Dolce-RC+.

Fig. 8. Comparison for Dolce-RC+ and DCQCN in a dynamic scenario.

i via two switches. All links have the capacity of 200 Gbps and 1 μs propagation delay. Long flows are mixed with bursty short flows. There are 24 senders which generate short flows following a Poisson arrival process to send 256 KB data during [0 ms, 150 ms]. The traffic load contributed by these short flows is about 10%. The number of long flows varies in two phases. Two long flows with size 1.1 GB are generated by two different senders at 0 ms. Six new long flows with size 230MB join at 75 ms. The message size is 8 KB.

Using the aforementioned topology and settings, we run DCQCN and Dolce-RC+ respectively. Fig. 8 presents the comparison for Dolce-RC+ and DCQCN in this scenario. We select two long flows and one short flow (representing the short flows generated by a specific sender) for illustration. As can be seen in Fig. 8(a), the short flows are always sent using high rates in DCQCN, significantly affecting the throughput of the two long flows and the aggregate throughput. Note that simply resuming from the rate used by the last short flow cannot help solve this issue, since the last rate is generally not optimal

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

across time-varying network environments. Dolce-RC+ is able to tackle this issue by dynamically selecting appropriate rates for the short flows via learning from the long-term online data. Fig. 8(c) shows that the sending rates of the short flows are properly suppressed in Dolce-RC+, inducing a much higher aggregate throughput. From Fig. 8(b) and Fig. 8(d), one can observe that Dolce-RC+ tends to yield lower MCTs and is able to complete all the flows much earlier, compared to DCQCN.

VII. Conclusion

In this paper, we have presented a data-driven rate control framework for RDMA networks, which can learn from long-term online data via lightweight MAB online learning. Based on the framework, we proposed two rate control schemes, namely Dolce-RC and Dolce-RC+, which dynamically control the sending rate by learning from online data. Dolce-RC has been implemented in commodity smart NICs. Extensive testbed and simulation results have demonstrated the superiority of Dolce-RC over DCQCN. Dolce-RC+ is a clean-slate design which shows a promising purely data-driven new congestion control paradigm for next generation of DCNs.

Appendix A
Proof of Theorem 1

Proof. For arm i, let $\mu_i(t)$ be the expectation of the reward $X_i(t)$. Let i_t^* be the arm with highest expected reward at time t. Let I_t be the arm selected by an MAB algorithm π at time t. The regret of the MAB algorithm π is defined as follows:

$$\mathrm{Reg}_\pi(T) = \mathbb{E}[\sum_{t=0}^{T} X_{i_t^*}(t) - \sum_{t=0}^{T} X_{I_t}(t)]. \tag{6}$$

The first summation term in (6) is total rewards of the optimal algorithm which can track the best arm i_t^* at every time t.

Let $N_i(T) = \sum_{t=0}^{T} \mathbb{1}(I_t = i \neq i_t^*)$ be the number of times that the non-best arm i has been selected at time T. Let $\mathbb{E}_{\gamma,\epsilon}$ be the expectation under our DR-UCB algorithm using the discount factor γ with the random exploration probability ϵ. The regret in (6) can be rewritten as

$$\mathrm{Reg}_\pi(T) = \mathbb{E}_{\gamma,\epsilon}[\sum_{t=0}^{T} \sum_{i:\mu_i(t)<\mu_{i_t^*}(t)} (X_{i_t^*}(t) - X_{I_t}(t))\mathbb{1}(I_t = i)] \tag{7}$$

$$\leq \sum_{t=0}^{T} \mathbb{E}_{\gamma,\epsilon}[\sum_{i=0}^{K-1} \mathbb{1}(I_t = i \neq i_t^*)] \tag{8}$$

$$\leq \sum_{i=0}^{K-1} \mathbb{E}_{\gamma,\epsilon} N_i(T). \tag{9}$$

Here, (8) is based on that the rewards are bounded and (9) is derived by the linearity of expectation and definition of $N_i(T)$.

Let $\mathbb{E}_\gamma$ be the expectation excluding the random exploration. According to Lines 3–7 of Algorithm 1, we have,

$$\sum_{i=0}^{K-1} \mathbb{E}_{\gamma,\epsilon} N_i(T) \leq (1-\epsilon) \sum_{i=0}^{K-1} \mathbb{E}_\gamma[N_i(T)] + \frac{K-1}{K}\epsilon T. \tag{10}$$

Note that (10) holds because the DR-UCB algorithm spends at most $(K-1)T/\lfloor \frac{K}{\epsilon} \rfloor$ times randomly exploring non-optimal arms and selects the arms based on the discounted UCB procedure in Line 7 of Algorithm 1 in the other $(1-\epsilon)T$ rounds. This discounted UCB procedure is adopted from the discounted UCB algorithm [30] and [16].

To analyze the regret caused by the discounted UCB procedure, we will directly make use of the tight bound of $\mathbb{E}_\gamma[N_i(T)]$ in the following lemma from [30].

Lemma 1 (Remark 3 in [30])**.** *Let C_T be the number of times that optimal arm changes in T iterations. Given the total number of iterations T and taking the discount factor $\gamma = 1 - \frac{\sqrt{C_T/T}}{4}$, then we have*

$$\mathbb{E}_\gamma[N_i(T)] = O(\sqrt{C_T T}\log T). \tag{11}$$

Assume that C_T is a constant. From (10) and (11), we have

$$\sum_{i=0}^{K-1} \mathbb{E}_{\gamma,\epsilon} N_i(T) \leq O(\sqrt{C_T T}\log T)\cdot(1-\epsilon)K + \epsilon T \tag{12}$$

$$= (1-\epsilon)O(\sqrt{C_T T K}\log T) + \epsilon T \tag{13}$$

$$= (1-\epsilon)O(\sqrt{T}) + \epsilon T. \tag{14}$$

Combining (9) and (14), we complete the proof of Theorem 1. □

Appendix B
Proof of Theorem 2

Proof. We make use of the fluid model of DCQCN in [7] and incorporate two major changes into it as follows.

First, we incorporate (1) with the new parameter $\beta^{(i)}$ for connection i into the fluid model of DCQCN such that the differential equation for the evolution of the target rate $R_T^{(i)}$ of connection i is modified to the following equation:

$$\frac{dR_T^{(i)}}{dt} = -\frac{R_T^{(i)}(t) - \beta^{(i)} R_C^{(i)}(t)}{\tau}\left(1 - \left(1-p(t-\tau^*)\right)^{\tau R_C^{(i)}(t-\tau^*)}\right) + R_{AI} R_C^{(i)}(t-\tau^*)\frac{\left(1-p(t-\tau^*)\right)^{FB} p(t-\tau^*)}{\left(1-p(t-\tau^*)\right)^{-B} - 1} + R_{AI} R_C^{(i)}(t-\tau^*)\frac{\left(1-p(t-\tau^*)\right)^{FTR_C^{(i)}(t-\tau^*)} p(t-\tau^*)}{\left(1-p(t-\tau^*)\right)^{-TR_C^{(i)}(t-\tau^*)} - 1}, \tag{15}$$

where τ is the CNP generation timer, R_{AI} is a fixed rate increase step in the additive increase phase, τ^* denotes the steady-state round-trip time (i.e., control loop delay), $p(t-\tau^*)$ represents the ECN-marking probability at time $(t-\tau^*)$, F is the fast recovery steps, B is the byte counter for rate increase, T is the timer for rate increase, and the variables with superscript (i) indicate that they are associated with connection i. Note that $\beta^{(i)}$ only appears in the first term of (15) since

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.

it only controls the relation between the target rate and the current rate in the fast recovery phase.

Second, we remove the differential equation for the evolution of the rate reduction factor (i.e., the equation (5) in [7]) from the original fluid model of DCQCN, such that the differential equation for the evolution of the current rate $R_C^{(i)}$ of connection i is revised to the following equation:

$$\begin{aligned}\frac{dR_C^{(i)}}{dt} =& -\frac{R_C^{(i)}(t)\alpha^{(i)}}{2\tau}\left(1-\left(1-p(t-\tau^*)\right)^{\tau R_C^{(i)}(t-\tau^*)}\right)\\ &+\frac{R_T^{(i)}(t)-R_C^{(i)}(t)}{2}\frac{R_C^{(i)}(t-\tau^*)p(t-\tau^*)}{\left(1-p(t-\tau^*)\right)^{-B}-1}\\ &+\frac{R_T^{(i)}(t)-R_C^{(i)}(t)}{2}\frac{R_C^{(i)}(t-\tau^*)p(t-\tau^*)}{\left(1-p(t-\tau^*)\right)^{-TR_C^{(i)}(t-\tau^*)}-1},\end{aligned} \tag{16}$$

where the parameter $\alpha^{(i)}$ for connection i is now a constant and does not depend on time.

Based on the revised DCQCN fluid model with (15) and (16) (i.e., the fluid model for Dolce-RC), we analyze the equilibrium (i.e., steady-state) of the resulting system, so as to derive the relation between α and β for Dolce-RC in the steady state.

To simplify the analysis, we utilize the same auxiliary variables a, b, c, d, and e as that used in [7].

The equilibrium point of the system can be derived by setting the left-hand sides of all the differential equations in the fluid model to zero and ignoring the time index t. Specifically, setting the left-hand side of (15) to zero yields:

$$-\frac{R_T^{(i)*}-\beta^{(i)}R_C^{(i)*}}{\tau}a+R_{AI}R_C^{(i)*}c+R_{AI}R_C^{(i)*}e=0, \tag{17}$$

where $R_T^{(i)*}$ denotes the steady-state target rate of connection i.

Similarly, setting the left-hand side of (16) to zero gives:

$$-\frac{R_C^{(i)*}\alpha^{(i)}}{2\tau}a+\frac{R_T^{(i)*}-R_C^{(i)*}}{2}R_C^{(i)*}b+\frac{R_T^{(i)*}-R_C^{(i)*}}{2}R_C^{(i)*}d=0. \tag{18}$$

From (18), we can further derive:

$$R_T^{(i)*}=R_C^{(i)*}+\frac{\alpha^{(i)}a}{\tau(b+d)}. \tag{19}$$

Substituting (19) into (17), we can obtain:

$$\alpha^{(i)}=\frac{\beta^{(i)}-1+\frac{\tau R_{AI}}{a}(c+e)}{\frac{a}{\tau R_C^{(i)*}(b+d)}}, \tag{20}$$

which is identical to (3).

□

ACKNOWLEDGMENT

The work of Kechao Cai was supported in part by NSF China under Grant 62202508 and in part by Shenzhen Science and Technology Program (Grant No. 202206193000001, 20220817094427001). The work of John C.S. Lui was supported in part by the RGC SRFS2122-4S02.

REFERENCES

[1] Infiniband Trade Association, "InfiniBand architecture specifications," volume 1, release 1.5, 2021.
[2] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," *Proc. of ACM SIGCOMM*, 2015.
[3] Infiniband Trade Association, "Supplement to InfiniBand architecture specification," volume 1, release 1.2.2, annex A17: RoCEv2 (IP routable RoCE), 2014.
[4] IEEE. 802.11Qbb, "Priority-based flow control," 2011.
[5] IEEE. 802.11Qau, "Congestion notification," 2010.
[6] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," *Proc. of ACM SIGCOMM*, 2010.
[7] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "ECN or delay: Lessons learnt from analysis of DCQCN and TIMELY," *Proc. of ACM CoNEXT*, 2016.
[8] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan, and Y. Wang, "Aeolus: A building block for proactive transport in datacenters," *Proc. of ACM SIGCOMM*, 2020.
[9] P. Goyal, P. Shah, K. Zhao, G. Nikolaidis, M. Alizadeh, and T. E. Anderson, "Backpressure flow control," *Proc. of USENIX NSDI*, 2022.
[10] HPC-AI Advisory Council. https://www.hpcadvisorycouncil.com.
[11] E. Hazan, "Introduction to online convex optimization," *Found. and Trends in Optimization*, Vol. 2, No. 3-4, pp. 157–325, 2016.
[12] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," *Proc. of USENIX NSDI*, 2015.
[13] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, P. B. Godfrey, and M. Schapira, "PCC Vivace: Online-learning congestion control," *Proc. of USENIX NSDI*, 2018.
[14] X. Nie, Y. Zhao, Z. Li, G. Chen, K. Sui, J. Zhang, Z. Ye, and D. Pei, "Dynamic TCP initial windows and congestion control schemes through reinforcement learning," *IEEE Journal on Selected Areas in Communications*, Vol. 37, No. 6, pp. 1231–1247, 2019.
[15] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, Vol. 47, No. 2-3, pp. 235–256, 2002.
[16] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," *Proc. of International Conference on Algorithmic Learning Theory*, 2011.
[17] C. S. Turner, "A fast binary logarithm algorithm," *IEEE Signal Processing Magazine*, pp. 124–140, 2010.
[18] J. L. Maire, N. Brunie, F. Dinechin, and J.-M. Muller, "Computing floating-point logarithms with fixed-point operations," *Proc. of IEEE Symposium on Computer Arithmetic*, 2016.
[19] Fixed-point base 2 logarithm algorithm implementation in C. https://github.com/dmoulding/log2fix.
[20] M. Istoan and B. Pasca, "Fixed-point implementations of the reciprocal, square root and reciprocal square root functions," 2015.
[21] Network simulator 3. https://www.nsnam.org/.
[22] Ns-3 simulator for RDMA. https://github.com/bobzhuyb/ns3-rdma.
[23] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "HPCC: High precision congestion control," *Proc. of ACM SIGCOMM*, 2019.
[24] MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE. https://mvapich.cse.ohio-state.edu/benchmarks/.
[25] The Vienna Ab initio Simulation Package: Atomic scale materials modelling from first principles. https://www.vasp.at.
[26] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," *Proc. of ACM SIGCOMM*, 2015.
[27] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," *Proc. of ACM SIGCOMM*, 2009.
[28] NVIDIA's GPU VASP benchmarks. https://github.com/smaintz-nv/gpu-vasp-files/tree/master/benchmarks/silicaIFPEN.
[29] N. Bansal and M. Harchol-Balter, "Analysis of SRPT scheduling: Investigating unfairness," *Proc. of ACM SIGMETRICS*, 2001.
[30] A. Garivier and E. Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on June 26,2024 at 05:12:35 UTC from IEEE Xplore. Restrictions apply.